# Name : Shahzaib Khan(67421)

# OS Resource Manager Module

## 1. Introduction

The growing complexity of educational computing environments requires robust OS-level solutions to manage limited resources efficiently. This project aims to simulate an operating system's core resource management components tailored for a university lab setting. The simulator handles memory management, file operations, disk I/O scheduling, and user-level security policies. This modular implementation is written in C and mimics real-world OS behavior to support concurrent access by multiple student processes while enforcing access control and maintaining system stability.

## 2. Literature Review

Operating systems are responsible for the effective management of hardware and software resources. Literature shows that traditional OS components—such as memory paging (Silberschatz et al., 2020), file system allocation methods (Tanenbaum, 2015), and disk scheduling algorithms (Galvin, 2018)—are fundamental for multi-user, multi-process environments.

Key concepts:

* • Paging Systems help efficiently allocate memory using fixed-size blocks, reducing fragmentation.
* • File Systems use hierarchical structures and allocation techniques (contiguous, linked, indexed) for efficient file access.
* • Disk Scheduling improves I/O performance using algorithms like FCFS, SSTF, SCAN, and LOOK.
* • Security Models implement role-based access controls (RBAC) to enforce file-level protection and isolation.

These models served as the theoretical foundation for designing the OS Simulator.

## 3. System Design & Role Chart

📐 Architecture Diagram (Describe or sketch):

memory\_manager.c: Simulates paging, page faults, replacement policy.

file\_system.c: Manages files, directories, access modes, allocation.

disk\_scheduler.c: Handles virtual memory disk requests.

security\_module.c: Enforces user role and file access policy.

🧾 User Roles:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Role | Read | Write | Delete | Execute |
| Admin | ✅ | ✅ | ✅ | ✅ |
| Student | ✅ | ✅ | ❌ | ❌ |
| Guest | ✅ | ❌ | ❌ | ❌ |

## 4. Methodology

This simulator was developed on Windows using MSYS2 (with gcc and make) but is portable to Linux (Ubuntu/Fedora). Development followed a modular structure with each component in its own .c/.h pair.

🛠 Technologies:

* • Language: C
* • Platform: MSYS2 (Windows) / Linux compatible
* • Compiler: GCC

Phases:

* • Memory Management: Implemented FIFO-based paging with page fault logging.
* • File Management: Created a hierarchical system supporting read/write/delete based on permissions.
* • Disk Scheduling: FCFS, SSTF, and SCAN algorithms to simulate head movements.
* • Security Module: Role-based access control with activity logging.

## 5. Source Code

Include the source files here:

* • main.c
* • memory\_manager.c/.h
* • file\_system.c/.h
* • disk\_scheduler.c/.h
* • security\_module.c/.h
* • Makefile

You can attach these files or paste the critical parts (main logic) with comments.

## 6. Results and Discussion

✅ Simulation Results:

Memory:  
  
FIFO replacement  
  
7 Page Faults for test run

File System:  
  
Successfully blocked guest from writing or deleting  
  
Read-only protection enforced

Disk I/O:  
  
FCFS Head Movement: 640  
  
SSTF Head Movement: 236  
  
SCAN Head Movement: 169

Security Logs:  
  
Guest write and delete operations denied  
  
All operations logged with timestamp and role

📊 Discussion:  
  
The simulator performed as expected. While simplistic compared to real operating systems, the module accurately mimics core OS resource behavior, which is beneficial for educational purposes. Logging and modularity also support extensibility (e.g., adding LRU or priority-based disk I/O).